CLAIMS

1. A process for routing packets through a load balancing array of servers across a network in a computer environment, comprising the steps of:

providing a plurality of load balancing servers;

providing at least one back end Web server;

wherein one of said load balancing servers is also a scheduler;

wherein a request packet from a client is routed through said scheduler;

wherein said scheduler routes and load balances said request packet to a load balancing server;

wherein said load balancing server routes and load balances said request packet to a back end Web server;

wherein said back end Web server's response packet to said request packet is sent to said load balancing server; and

wherein said load balancing server sends said response packet directly to said client.

2. The process of Claim 1, wherein said scheduler routes and load balances client requests to itself.

3. The process of Claim 1, further comprising the steps of:

detecting the failure of said scheduler; and

electing one of said load balancing servers as the new scheduler.

4. The process of Claim 1, wherein said scheduler detects the failure of other load balancing servers; and wherein said scheduler stops routing packets to any failed load balancing servers.

5. The process of Claim 1, wherein said load balancing server schedules sessions to back end Web servers based on a cookie or session ID.

6. The process of Claim 1, wherein said load balancing server uses cookie injection to map a client to a specific back end Web server.

7. The process of Claim 1, wherein said load balancing server decrypts said request packet if it is an SSL session before routing and load balancing said request packet to a back end Web server.

8. The process of Claim 7, wherein said load balancing server encrypts said response packet if it is an SSL session before sending said response packet directly to said client.

9. The process of Claim 1, wherein said load balancing server establishes a connection with said client and said client keeps said connection alive with said load balancing server.

10. The process of Claim 9, wherein said load balancing server performs URL based scheduling of request packets.

11. The process of Claim 9, wherein said load balancing server performs hash scheduling of request packets.

12. The process of Claim 1, wherein said load balancing server maintains persistent connections in all its paths when required; and wherein said load balancing server uses hash group based persistence to maintain its persistence tables.

13. The process of Claim 1, wherein said load balancing server detects if a back end Web server fails; and wherein said load balancing server stops routing request packets to failed back end Web servers.

14. The process of Claim 1, further comprising the step of:
providing a content delivery network; and

wherein said load balancing server modifies select URLs in the HTML page in said response packet to serve them from said content delivery network.

15. The process of Claim 14, wherein HTML pages that have modified URLs are cached to improve performance.

16. An apparatus for routing packets through a load balancing array of servers across a network in a computer environment, comprising:

a plurality of load balancing servers;

at least one back end Web server;

wherein one of said load balancing servers is also a scheduler;

wherein a request packet from a client is routed through said scheduler;

wherein said scheduler routes and load balances said request packet to a load balancing server;

wherein said load balancing server routes and load balances said request packet to a back end Web server;

wherein said back end Web server's response packet to said request packet is sent to said load balancing server; and

wherein said load balancing server sends said response packet directly to said client.

17. The apparatus of Claim 16, wherein said scheduler routes and load balances client requests to itself.

18. The apparatus of Claim 16, further comprising:

a module for detecting the failure of said scheduler; and

a module for electing one of said load balancing servers as the new scheduler.

19. The apparatus of Claim 16, wherein said scheduler detects the failure of other load balancing servers; and wherein said scheduler stops routing packets to any failed load balancing servers.

20. The apparatus of Claim 16, wherein said load balancing server schedules sessions to back end Web servers based on a cookie or session ID.

21. The apparatus of Claim 16, wherein said load balancing server uses cookie injection to map a client to a specific back end Web server.

22. The apparatus of Claim 16, wherein said load balancing server decrypts said request packet if it is an SSL session before routing and load balancing said request packet to a back end Web server.

23. The apparatus of Claim 22, wherein said load balancing server encrypts said response packet if it is an SSL session before sending said response packet directly to said client.

24. The apparatus of Claim 16, wherein said load balancing server establishes a connection with said client and said client keeps said connection alive with said load balancing server.

25. The apparatus of Claim 24, wherein said load balancing server performs URL based scheduling of request packets.

26. The apparatus of Claim 24, wherein said load balancing server performs hash scheduling of request packets.

27. The apparatus of Claim 16, wherein said load balancing server maintains persistent connections in all its paths when required; and wherein said load balancing server uses hash group based persistence to maintain its persistence tables.

28. The apparatus of Claim 16, wherein said load balancing server detects if a back end Web server fails; and wherein said load balancing server stops routing request packets to failed back end Web servers.

29. The apparatus of Claim 16, further comprising:

a content delivery network; and

wherein said load balancing server modifies select URLs in the HTML page in said response packet to serve them from said content delivery network.

30. The apparatus of Claim 29, wherein HTML pages that have modified URLs are cached to improve performance.